

第十八节：基本数据结构

学习目标：

- 了解四种基本数据关系：集合、线性关系、树形关系、图形关系
- 理解几种常见的抽象数据结构：链表、栈、队列、树、图
- 掌握日常生活中数据结构的应用
- 理解不同数据结构在添加、删除、访问/查出/修改等方面的不同特性
- 重点理解数据的逻辑结构、数据的关系

0. 问题引入

学校举行集体舞比赛，如果你是班级的文艺委员，你该如何设计各种集体队形，让班级舞蹈更具变化与活力呢？在团队活动或项目中，人和人之间的连接关系有无穷多种变化，你能举出一些具体的例子来吗？



图 1a

篮球比赛



图 1b 做操队形



图 1c 花样游泳队形 1



图 1d 花样游泳队形 2

【问题思考】请想一想上述每一种不同情况下，人和人之间的连接有什么特点呢？

【问题引入 2】

三十年，我们的父辈通过书信和远在他乡的朋友们交流，他们是笔友。现在，我们可以在几秒钟内和地球另一端的外国朋友们视频交流，我们是网友。当今社会已经进入信息高度发达与联通的信息社会，信息的传递已经越来越不依赖人与人之间的物理距离，无论身在世界何处，都可以通过“网络”实时交流。下图是一张典型的因特网可视化表示，这样一张巨大的网络里面，每一个点都是一台计算机，我们将其作为一个数据点，那么各个点之间有哪些可能的连接关系呢？

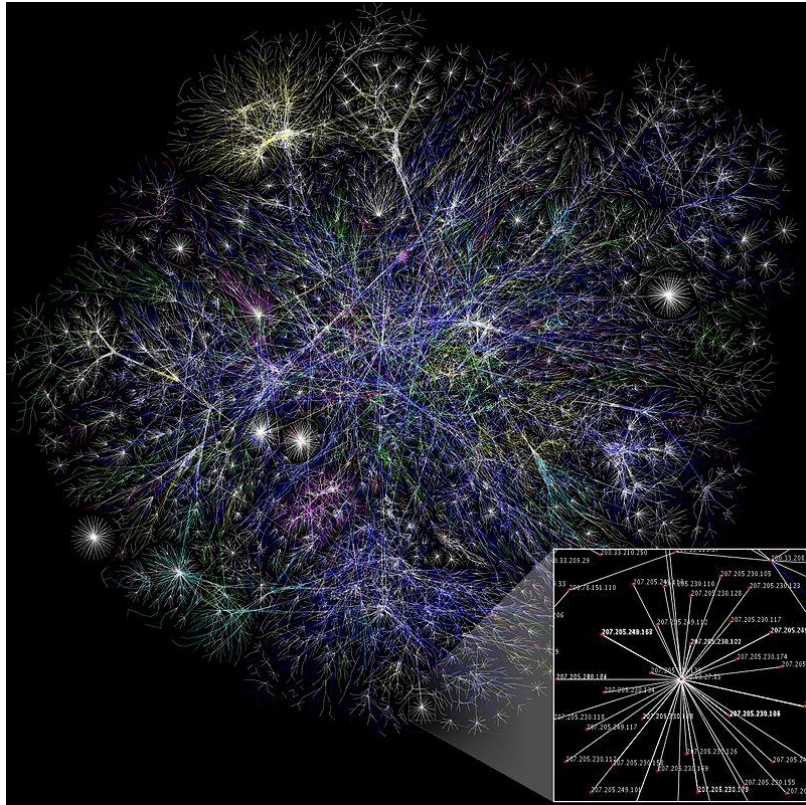
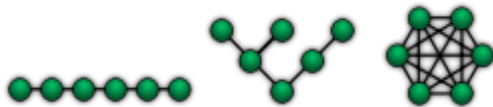


图 1 计算机网络可视化表示

https://en.wikipedia.org/wiki/File:Internet_map_1024.jpg

Partial map of the Internet based on the January 15, 2005 data found on opte.org. Each line is drawn between two nodes, representing two IP addresses. The length of the lines are indicative of the delay between those two nodes. This graph represents less than 30% of the Class C networks reachable by the data collection program in early 2005.

【问题思考 2】请尝试画出数据点之间可能的连接关系图。



1. 基本数据关系

要刻画并应用数据结构，我们要理解数据的逻辑结构和物理结构。逻辑结构是面向问题的；物理结构是面向计算机的，其基本的目标是将数据及其逻辑关系存储到计算机内存

中。¹ 本节我们并不考虑不同的数据结构如何在计算机中实现，而是主要关注数据的逻辑结构，也即数据元素之间的相互关系。

数据元素之间有四种基本相互关系：集合关系、线性关系、树形关系、图形关系，如图 2 所示。

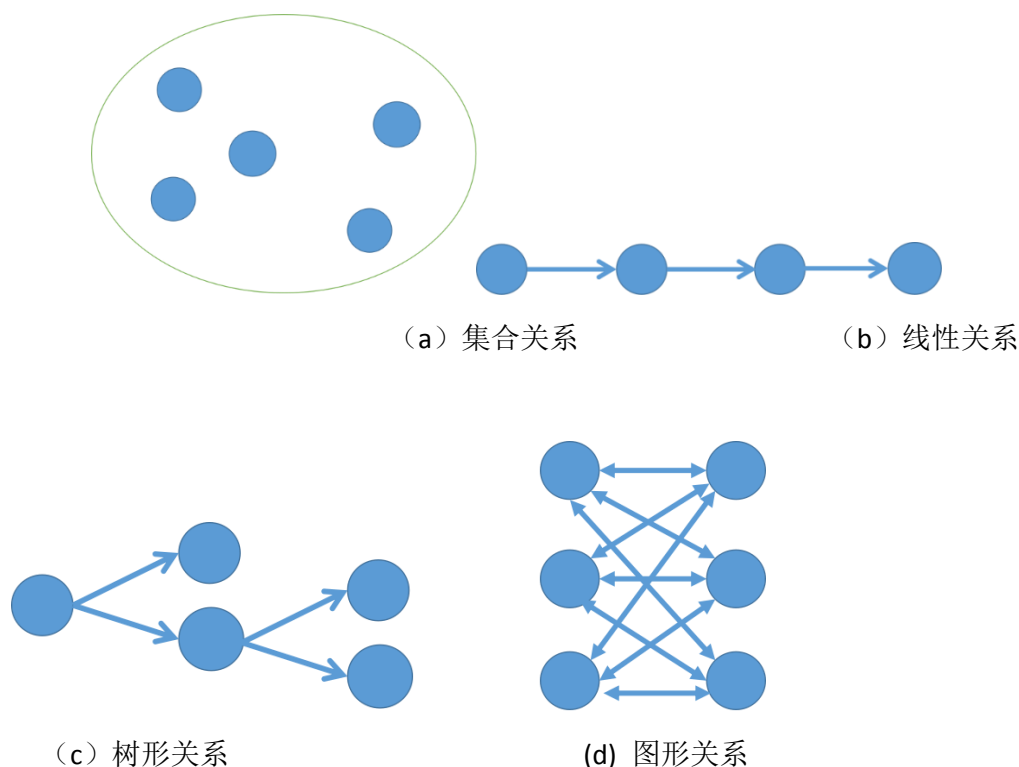


图 2 四种基本数据关系：集合关系、线性关系、树形关系、图形关系

集合关系：数据元素除了同属于一个集合之外，互相之间没有任何关系。它只考虑数据,不考虑关系.是一种松散的数据结构，如图 2a。例如，图 1a 中篮球场上的十名球员，他们同在球场上，但是互相之间没有连接关系，在球场上的位置可以任意变化。

线性关系：数据元素之间是一一对应的关系，如图 2b。一对一的结构,是最简单的结构.它只有一个没有前驱、只有后继的结点,叫着首结点；只有一个没有后继、只有前驱的结点,叫着尾结点；其余的结点都只有一个直接前驱和一个直接后继。例如，图 1b 中的做操直线队形，每个同学之间按照高矮顺序排列；再如军训站队列，从左向右报数，每个同学听其左侧同学报数，而向其右侧同学报下一个数。

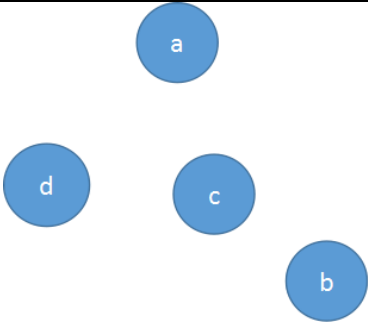
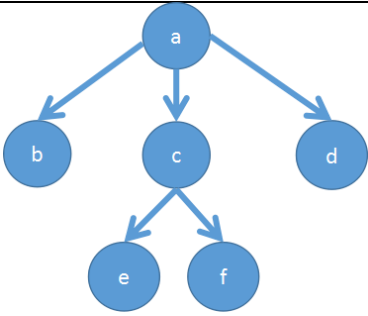
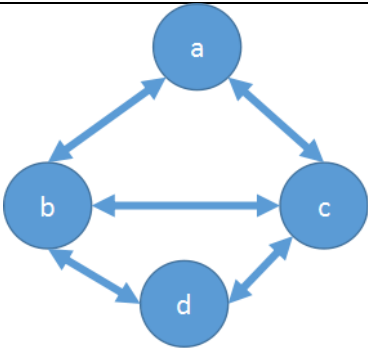
树形关系：数据元素之间是一对多的关系，如图 2c。是比较复杂的非线性结构.它只有一个没有前驱、只有后继的结点,叫着根结点；可有多个没有后继、只有前驱的结点,叫着叶子结点；其余的结点都只有一个直接前驱和多个直接后继。例如，图 1c 中的花样游泳队形，从上向下，每一个选手向下对应可能多个队友。

图形关系：数据元素之间是多对多的关系，如图 2d。是更加复杂的非线性结构.它的每一个结点都可能多个直接前驱和多个直接后继.其关系既可以是单向的,也可

¹ <http://blog.csdn.net/dingxingmei/article/details/19070037>

以是双向的。例如，图 1d 中的花样游泳队形，队员之间是多人互相连接，每个队员向上：手臂与多名队友互相连接，每个队员向下：腿与多名队友相互连接，是多对多的关系。

【练习】 请在下列表格中空白单元格处添加对应信息。

数据关系	元素之间逻辑关系	图形示例
集合关系		
线性关系		
	一对多	
图形关系		

注： 物理结构：是指数据的逻辑结构在计算机中的存储形式。主要有两种存储方式：

1. 顺序存储结构 2. 链式存储结构.

1：顺序存储结构：是把数据元素存放在地址连续的存储单元里，其数据的逻辑关系和物理关系是一致的。

2 链式存储结构：是把数据元素放在任意存储单元里，这组存储单元可以连续的，也可以是不连续的。数据元素的存储关系并不能反映其逻辑关系，因此需要用一个指针存放数据元素的地址，通过地址就可以找到相关联数据元素的位置。链式存储相比较顺序存储要

更为灵活。数据存在哪个具体的物理内存并不重要，重要的是有能找到该内存地址的线索。

2. 常用数据结构简介

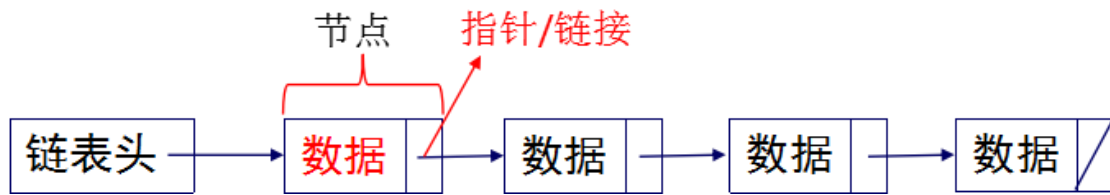
本节我们通过生活中的实例简单介绍几种常见的数据结构，并针对添加、删除、访问/查出/修改等数据操作说明各数据结构的特点。请同学们边学习，边思考记录，并填写完成下表，形成自己的理解。

数据关系	数据结构	生活实例	逻辑结构	数据关系	添加	删除	访问、修改、查找
线性	链表						
	栈						
	队列						
非线性	树						
	图						

2.1 链表

线性结构是最简单而应用最广泛的一种数据结构，在不同的场合会采取不同的存储结构和实现方法。线性表的两种实现方法，即顺序表和链表。顺序表是把数据元素存放在地址连续的存储单元里，其数据的逻辑关系和物理关系是一致的。链表：是把数据元素放在任意存储单元里，这组存储单元可以连续的，也可以是不连续的。数据元素的存储关系并不能反映其逻辑关系，因此需要用一个指针存放数据元素的地址，通过地址就可以找到相关联数据元素的位置。链表的存储相比较顺序表的存储要更为灵活。数据存在哪个具体的物理内存并不重要，重要的是有能找到该内存地址的指针（或称为链接）。

假设大家玩一种寻宝游戏，在游戏开始之初，你知道宝贝的地理坐标，当你到达该地点后，你会找到该宝贝及下一个宝贝所在位置的地理坐标，再按照该坐标前往下一个地点，直到找到所有宝贝。我们通过这个例子来类比链表的结构和逻辑，如图 3 所示。游戏开始时所知道的第一个宝贝的地理位置就是链表头。每到达一个地方，我们获取两样东西， 1）宝贝，2）下一个宝贝所在位置，这两样东西组成了链表的节点，其中宝贝对应于链表节点的数据，下一个宝贝所在位置对应于链表节点的指针或称为链接。当达到最后一个宝贝所在地时，这个节点就只有宝贝（数据），而没有下一宝贝所在位置（指针）。



【动手操作】拿出自己的A4活页夹，做如下操作：

添加：将今天的课堂教案打孔插入到上周信息技术科目笔记页后面。

删除：讲上次课的课堂教案页从活页本中取出。

修改：找到大数据最后一节的课堂教案与笔记，对其中的内容进行一处修改

回形针游戏：形成链表，请自己尝试添加、删除、修改操作。

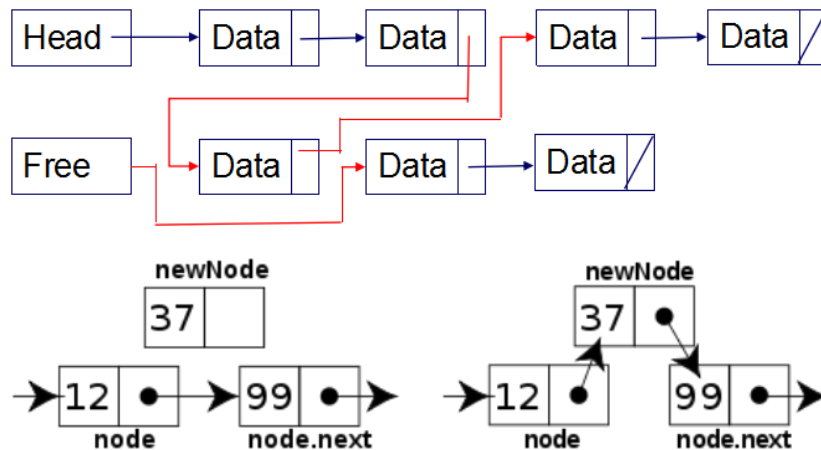


图 4 链表的插入

链表的插入：只需修改链表节点的指针。

在第 i 个节点之后插入新节点 x ，图 4 为例，具体步骤如下：

- 断开第 i 个节点（数据为 12 的节点）指向第 $i+1$ 个节点（数据为 99 的节点）的指针。
- 修改第 i 节点指针（数据为 12 的节点）：使其指向新节点 x （数据为 37 的节点）。
- 修改新节点 x （数据为 37 的节点）的指针：使其指向第 $i+1$ 个节点（数据为 99 的节点）。

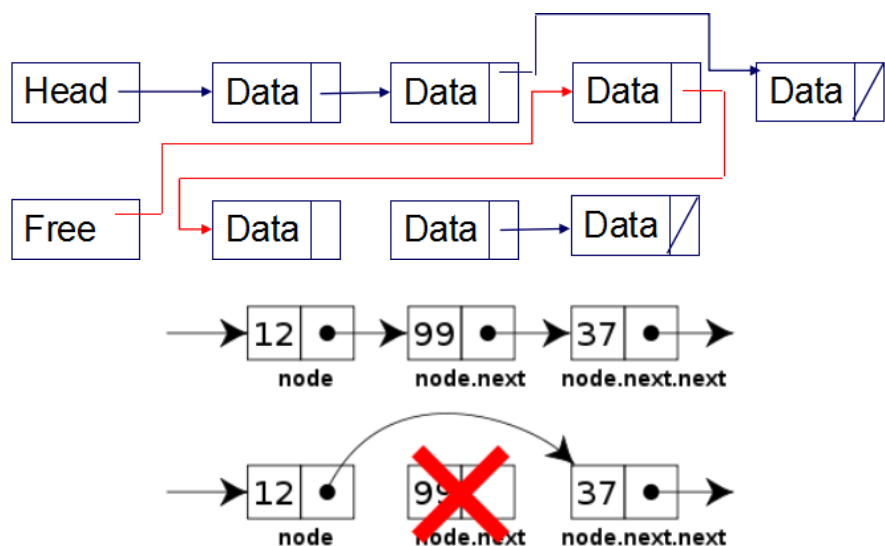


图 5 链表的删除

链表的删除：只需修改链表节点的指针。

删除第 i 个节点，图 5 为例，具体步骤如下：

- 断开第 i 个节点（数据为 99 的节点）指向第 $i+1$ 个节点（数据为 37 的节点）的指针。
- 修改第 $i-1$ 节点指针（数据为 12 的节点）：使其指向第 $i+1$ 个节点（数据为 37 的节点）。

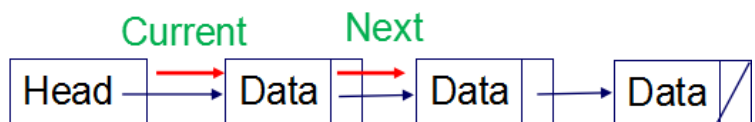


图 6 链表的查找/修改

链表的**查找/修改**：要查找第 i 个节点的数据，必须从表头开始依次访问各个节点，直到第 i 个节点。

2.2 栈

日常生活中，栈这种数据结构的例子很常见，例如：盘子洗好了，一个个往上摞，用的时候从最上面那个依次向下拿。

羊肉串：穿肉串——拿一根铁签，穿肉块，肉块从下往上依次穿上，如图 7a 所示；吃肉串——从上往下一块一块吃，如图 7b 所示，最后穿上的肉块是第一块吃到的；而最先穿上的肉块是最后一块吃到的。

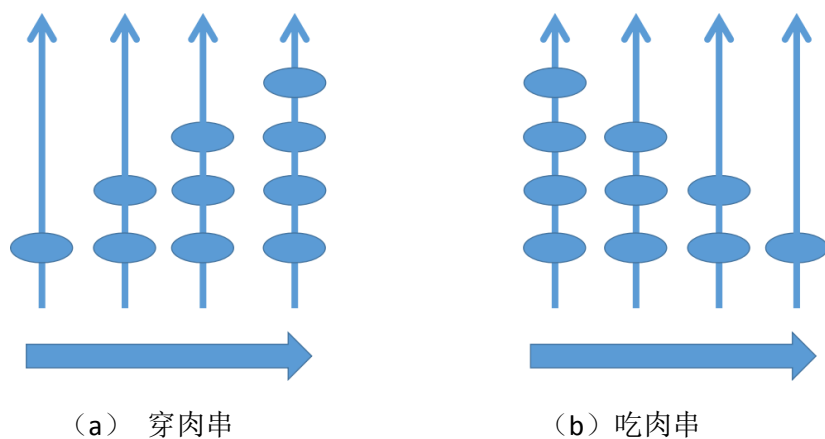


图 7 栈的羊肉串示意图

栈的概念：后进先出（先进后出）的数据结构就叫做栈，如图 8 所示，第一个进入栈的元素存在栈底，各元素按照进入先后顺序，一次擦上去，最后一个进入栈的元素在栈顶，使用一个栈顶指针始终指向栈顶元素，栈也是一种常见的线性数据结构。

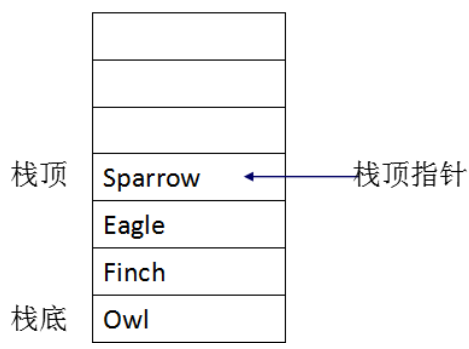


图 8 栈的表示

【问题思考】栈在生活中的例子，除了羊肉串、冰糖葫芦等，还有如图 9 中描述的一些例子，请你结合栈的特性思考一下，如何向栈添加或者删除元素？你还能举出生活中的其它例子吗？

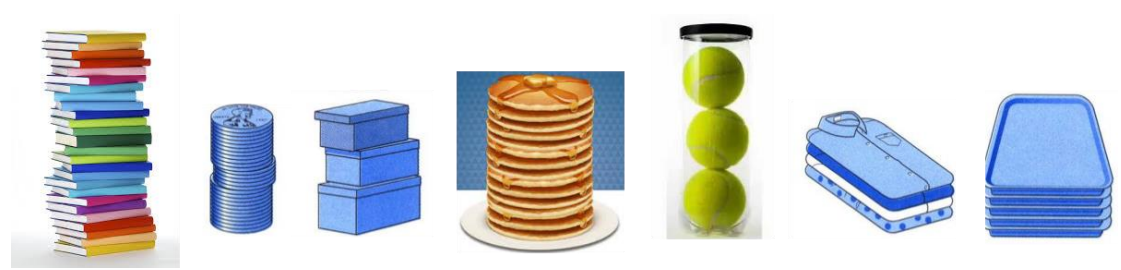


图 9 栈在生活中的实际例子

栈的添加：要向栈中添加/插入元素，只能添加在栈顶。栈的添加操作也叫作压入，如图 10 所示，具体方法为：

- 将栈顶指针向上移动一格
- 将新元素存入当前栈顶指针所指位置

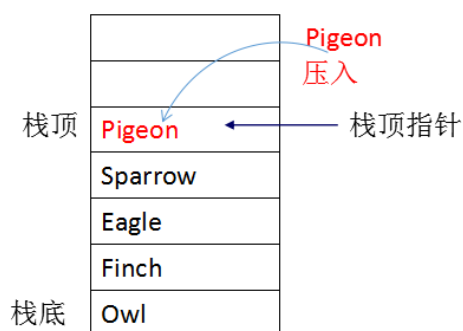


图 10 栈的添加(压入)示意图

栈的删除：要删除栈中的元素，只能删除栈顶元素。栈的删除操作也叫作弹出，如图 11 所示，具体方法为：

- 返回当前栈顶元素
- 将栈顶指针向下移动一格

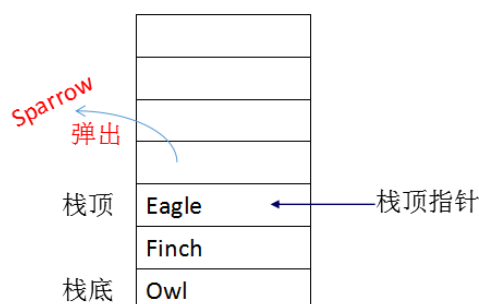


图 11 栈的删除(弹出)示意图

2.3 队列

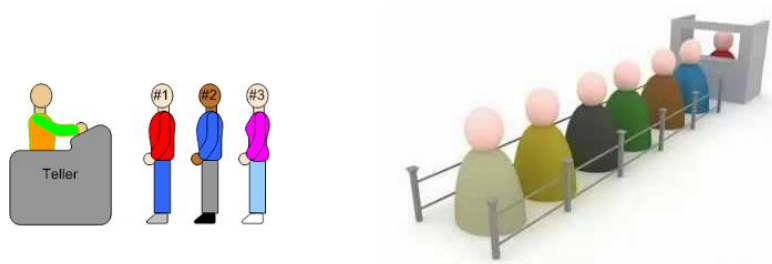


图 12 队列数据结构在生活中的实际例子

有的时候我们也许只需要在线性序列的一端进行操作，例如一摞盘子，我们只在盘子最上方拿、放盘子，这就是栈；但是有时候我们需要在一个线性序列的两端进行操作，例如，食堂排队的时候，你总是先找到队尾加入，而排在队首的同学打完饭之后就会离开，这时候栈就不再适用，我们需要引入一种新的数据结构，叫做队列，。这种数据结构顾名思义，来源与并应用于排队问题，如图 12，其数据元素符合“先进先出”、“后进后出”的规则，如图 13 所示。队列通过在线性序列上设定头指针和尾指针进行实现，如图 14 所示。

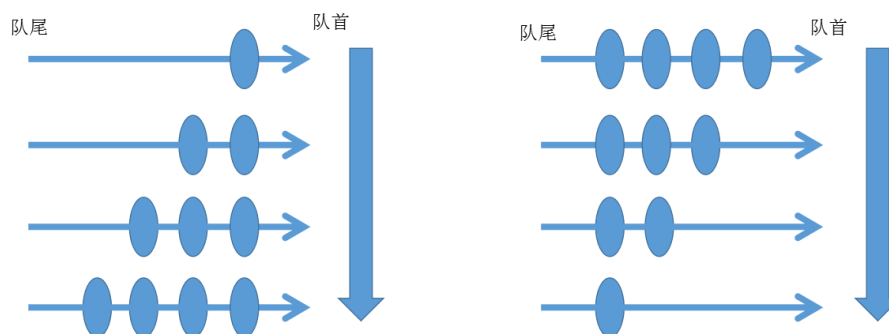


图 13 队列数据结构示意图

队列的添加：队列只能从尾部添加，如同排队，只能从队尾加入。

队列的删除：队列只能从头部删除，如同排队，完成事务后从队首离开。

队列这种数据结构能够用来建模并解决实际生活中的许多排队相关问题。

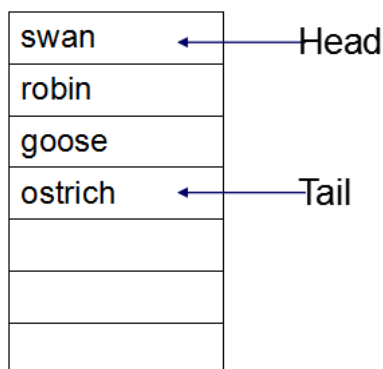


图 14 队列数据结构的实现方法

2.4 树

生活中，树形结构是很多客观事物与关系的抽象模型，例如亲属关系可以形成一个族谱，如图 15 所示，公司的上下级关系可以形成一个人事表。

树是一种非线性结构，元素之间的数据关系是一对多。树由三种不同类型的节点及其之间的连接构成，其与真实的生活中的树非常相似，如图 16 所示。1) “根”节点：每颗树有且只有一个根节点，对应于真实的树根。2) “叶”节点：没有子孙的节点，都叫做叶节点，对应于真实的树叶。3) 普通节点：不是根节点也不是叶节点的节点，对应于树枝。

树节点之间的连接：“根”节点与一定数量的普通节点以父-子关系联系起来，而其子节点也与另外的节点项以同样的方式联系。除了根节点的每个节点都有且只有一个父节点，并可能有一些子节点。如果没有子节点，那么就成为了“叶”节点。每个父节点左边的所有节点及其连接，也是一个树型结构，称作左子树，同理，每个父节点右侧的子树叫做右子树。如果每个节点（除了叶节点）都之多有两个子节点，那么这种树有个特殊的名字叫做“二叉树”。如果每个节点（除了叶节点）都刚好是两个子节点，就叫做“完全二叉树”。如图 17A，是一个深度为 3 的完全二叉树，图 17B 是一个

古代族谱，其构成一个深度为 4 的完全二叉树。下面我们都以二叉树为例来说明树的查找、添加等操作。

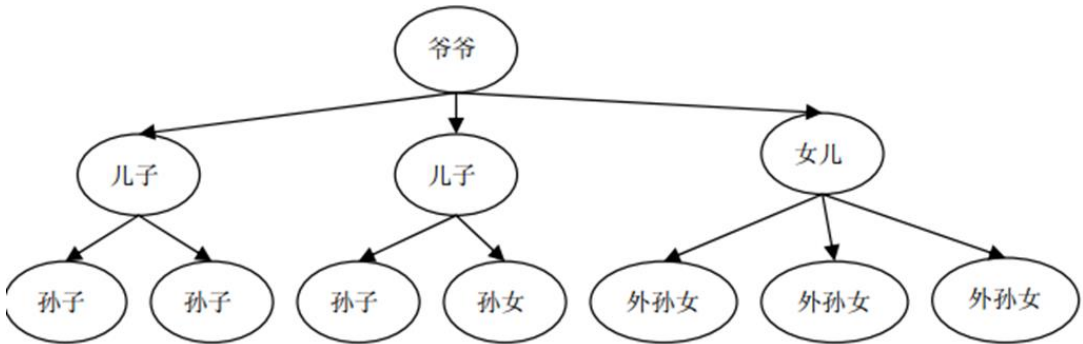


图 15 数据结构——树在生活中的实际例子：家谱

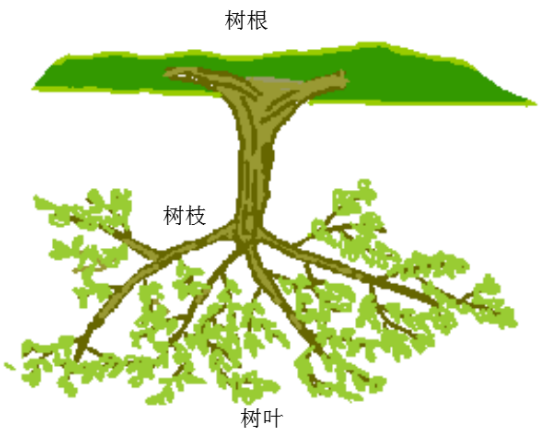
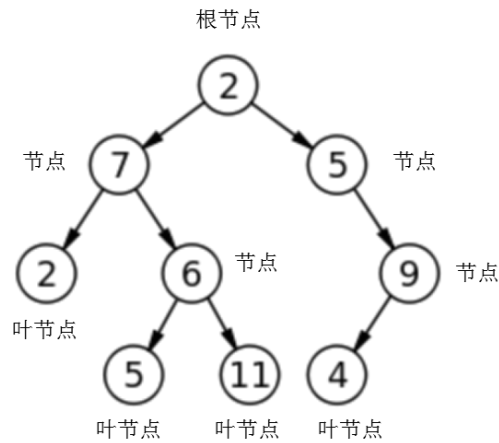


图 16 数据结构——树是一颗倒过来的树

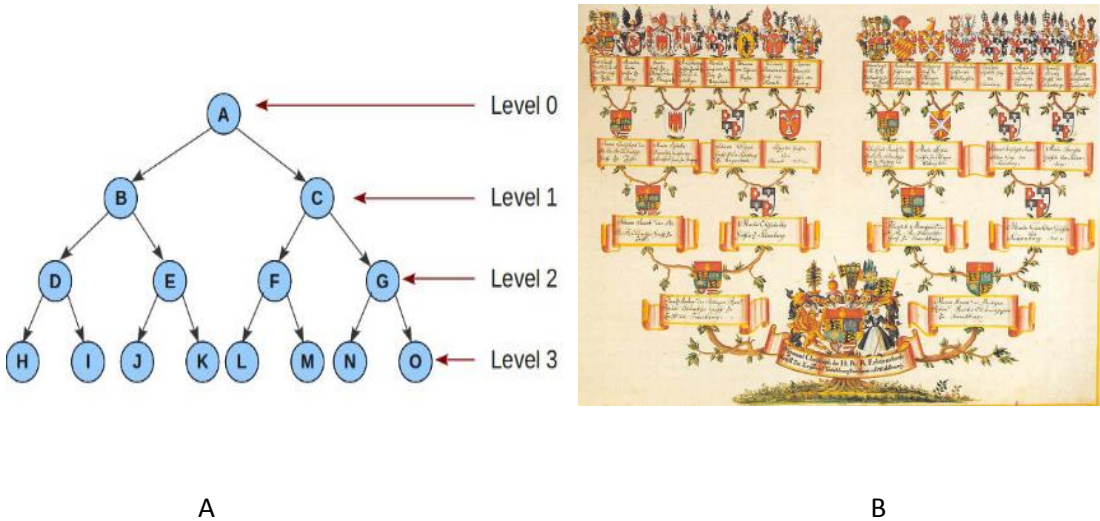


图 17 二叉树（完全二叉树）的表示及其实际例子。右图是一个四层族谱图。
树的查找：

你是不是发现，在线性表中查找一个元素每次都要遍历整个表？效率如此低下，怎么改进呢？树型结构就是最好的解决方案之一。请回忆一下上一节《算法的概念》中关于猜数字游戏的例子，所使用的二分法算法与树型结构有着非常密切的联系。例如在 1-10 之间猜数字，将 1-10 这十个数字构建为一个二叉树结构，要找的元素是 3，我们需要比较 4 次。具体算法为：1) 3 和 6 比较，由于 $3 < 6$ ，那么我们可以弃掉根节点的右子树。将搜索范围缩减为根节点的左子树。2) 3 和 4 比较，由于 $3 < 4$ ，那么我们可以弃掉“4”节点的右子树。将搜索范围缩减为“4”节点的左子树。3) 3 和 2 比较，由于 $3 > 2$ ，那么我们可以弃掉“2”节点的左子树。将搜索范围缩减为“2”节点的右子树。4) 3 和 3 比较， $3 = 3$ ，找到目标元素。算法结束。

简单来说，树非常适合查找操作，因为非常自然的每次比较后可以“砍掉”一半。

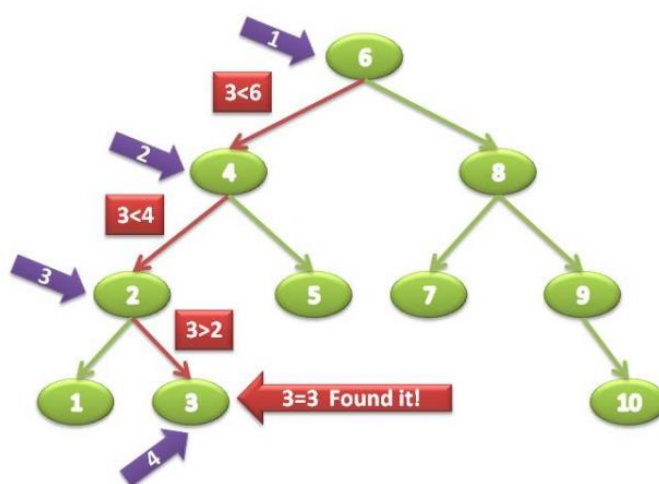


图 18 树的查找示意图

【问题思考】对于上节的猜数字游戏，如何用树形结构来实现呢？请你简单描述一下。

树的添加：树的添加/插入，与树的查找很相似，首先通过查找确定该添加的位置，然后在该叶节点处建立一个指向新节点的连接。如图 19 所示。

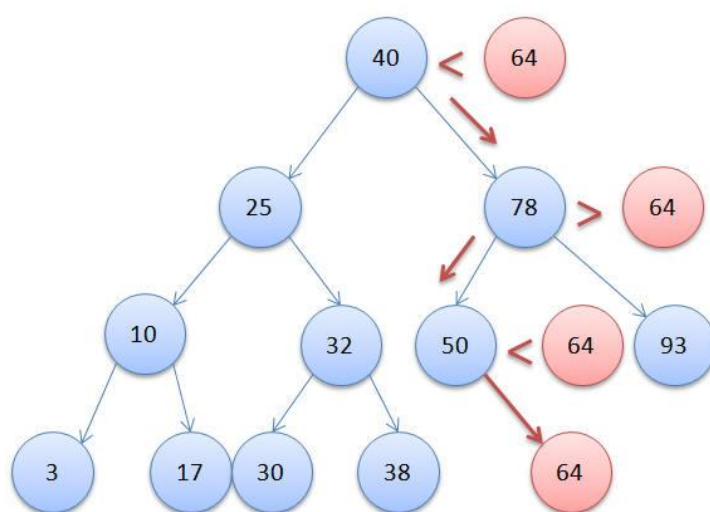


图 19 树的添加(插入)示意图

树的删除较为复杂，这里我们就不进行介绍了，感兴趣的同学可以自行查阅资料。

2.5 图

图是一种更为复杂的非线性数据结构，其数据元素之间呈现多对多的关系。在日常生活中的例子包括：通信网络、公路交通网、地铁网络、社交网络等。

图是由结点的有穷集合 V 和边的集合 E 组成，通过表示为 $G(V,E)$ ，其中， G 标示一个图， V 是图 G 中顶点的集合， E 是图 G 中边的集合。其中，为了与树形结构加以区别，在图结构中常常将结点称为顶点，边是顶点的有序偶对，若两个顶点之间存在一条边，就表示这两个顶点具有相邻关系。² 在图 20 的两个图结构中，一个是有向图，即每条边都有方向，另一个是无向图，即每条边都没有方向。权 (Weight)：有些图的边和弧有相关的数，这个数叫做权 (Weight)。这些带权的图通常称为网 (Network)。³

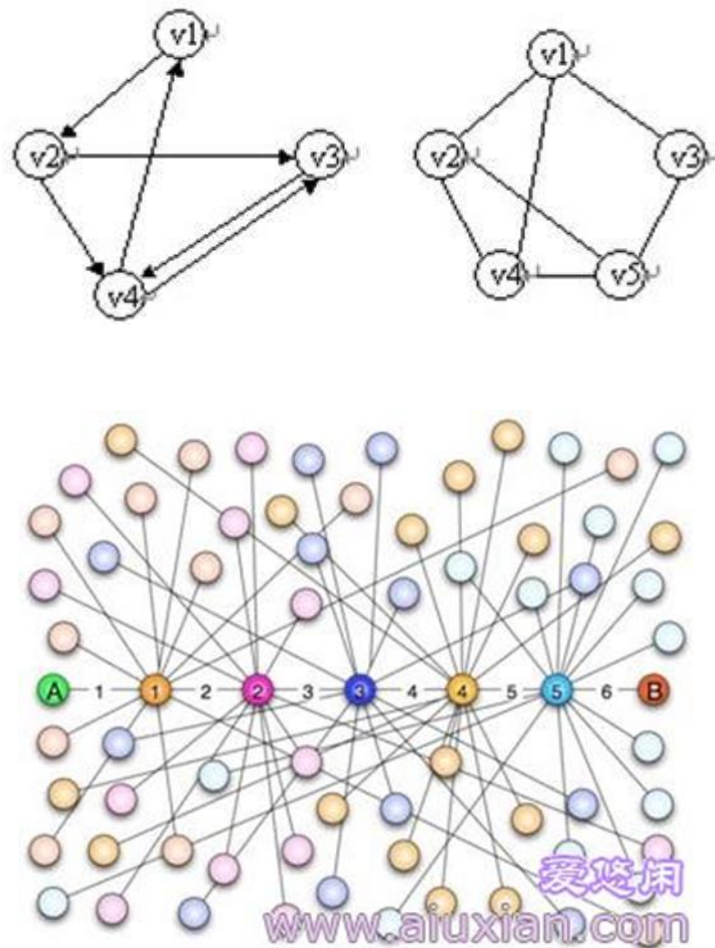


图 21 图在生活中的实例：六度空间理论

在图中，每个顶点都可以与一个或多个其它顶点联系起来，其中每个顶点都是平等的，类似于无根节点、无亲-子关系的树。图常常用于对实际问题进行建模，并解决这些问题。例如本节引言中的计算机网络，再比如社交网络的“六度空间”理论，又称作

² <http://www.cnblogs.com/begtostudy/archive/2010/09/20/1831962.html>

³ <http://www.cnblogs.com/w-wanglei/p/figure.html>

“六度分隔（Six Degrees of Separation）”理论。这个理论可以通俗地阐述为：“你和任何一个陌生人之间所间隔的人不会超过六个，也就是说，最多通过五个人你就能够认识任何一个陌生人。”如图 21 所示。

图在经典的旅行商问题、最短路径等经典为题中都起着核心作用。感兴趣的同学可以研究探索相关问题。

【练习】

1. 表演活动：请同学以小组为单位，自编自导用小短剧的方式将下面的内容表演出来，并录制视频。

- 链表的基本概念和原理，添加、删除、修改等操作特点及生活中的应用
- 栈的基本概念和原理，添加、删除等操作特点及生活中的应用
- 队列的基本概念和原理，添加、删除等操作特点及生活中的应用
- 二叉树的基本概念和原理，添加、查找等操作特点及生活中的应用
- 图的基本概念和原理及生活中的应用

2. 请将下列表格内容填写完整。

数据关系	数据结构	生活实例	逻辑结构	数据关系	添加	删除	访问、修改、查找
线性	链表	活页夹、链条、寻宝游戏	线性：一对一	每个节点由数据及指向下一个节点的指针组成	方便 仅修改节点连接	方便 仅修改节点连接	效率较低，需要遍历
	栈	羊肉串、糖葫芦、一叠盘子、一叠衣服、火车调度轨道	线性：一对一	后进先出，仅在线性结构的一端进行操作	压入 仅在栈顶	弹出 仅在栈顶	效率较低，需要遍历
	队列	排队	线性：一对一	先进先出，在线性结构的首尾两端进行操作	队尾	队首	效率较低，需要遍历

非线性	树	家谱、人事表	非线性：一对多	根节点、叶节点与普通节点之间的连接关系。二叉树。	基于二分搜索	<div><div></div><div></div><div></div></div>	二分搜索原理
	图	社交网络、计算机网络、交通图等	非线性：多对多	每个顶点都可以通过边与一个或多个其它顶点联系起来，其中每个顶点都是平等的。			